**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 4 з дисципліни

«Проектування алгоритмів»

„**Проектування і аналіз алгоритмів для вирішення NP-складних задач ч.1**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-12 Казаков Володимир Олексійович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.Н.*

Київ 2022

Графік залежності якості результату від кількості ітерацій:

Код на python:

import random

def generate\_things(min\_price, max\_price, min\_weight, max\_weight, things\_amount):

things = []

for i in range(things\_amount):

things.append([random.randint(min\_price, max\_price), random.randint(min\_weight, max\_weight)])

return things

def get\_sum(population, things):

sum = 0

i = 0

for p in population:

if p == 1:

sum = sum + things[i][1]

i = i + 1

return sum

def generate\_population(genes\_amount, things):

population = []

for i in range(5):

population.append([])

for j in range(genes\_amount):

if get\_sum(population[i], things) <= 250:

population[i].append(random.randint(0,1))

else:

population[i].append(0)

if get\_sum(population[i], things) > 250:

population[i].pop()

population[i].append(0)

return population

def get\_price(population, things):

sum = 0

for k in range(100):

if population[k] == True:

sum = sum + things[k][0]

return sum

def mutation(population, mutation\_chance):

chance = random.randint(0, 100)

if chance <= mutation\_chance:

first\_gene = random.randint(0, 99)

second\_gene = random.randint(0, 99)

buf = population[first\_gene]

population[first\_gene] = population[second\_gene]

population[second\_gene] = buf

return population

def best\_population(population, things):

index = 0

count = 0

for row in population:

if get\_price(row, things) >= get\_price(population[index], things):

index = count

count = count + 1

return index

Файл з основним кодом:

import random

import module1

bag\_size = 250

things\_amount = 100

population\_size = 5

things = module1.generate\_things(2, 30, 1, 25, things\_amount)

print('Generated array of things:')

for row in things:

print('Price:', row[0], 'Weight:', row[1])

population = module1.generate\_population(things\_amount, things)

print('\n',"Start population:")

for row in population:

print(row)

result = population[module1.best\_population(population, things)]

for i in range(1000):

first\_population = population[module1.best\_population(population, things)]

second\_population = population[random.randint(0, population\_size - 1)]

new\_population = []

for j in range(100):

if j < 30:

new\_population.append(first\_population[j])

else:

new\_population.append(second\_population[j])

new\_population = module1.mutation(new\_population, 5)

population.append(new\_population)

if module1.get\_sum(population[module1.best\_population(population, things)], things) <= 250:

result = population[module1.best\_population(population, things)]

if i % 20 == 0:

print("Iteration", i, "Weight =", module1.get\_sum(result, things), "Price =", module1.get\_price(result, things))

print(result)

print("New weight =", module1.get\_sum(new\_population, things), "New price =", module1.get\_price(new\_population, things))

print(result)

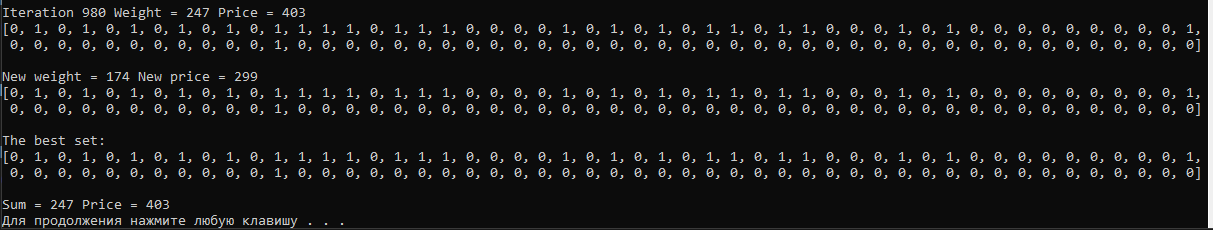
population\_size = population\_size + 1

print("The best set:")

print(result)

print("Sum =", module1.get\_sum(result, things), "Price =", module1.get\_price(result, things))

Приклад роботи програми:



Тут програма вивела результат ітерації №980, де показаний найкращий результат серед усіх наборів, та нижче набор, що був згенерований в результаті схрещування та можливої мутації. Над кожним з результатів показані вага та ціна.

Так як всього 1000 ітерацій, то ітерація №980 вивелась останньою, а після неї було виведено результат роботи всієї програми – тобто набір з найкращою ціною.